Drawing Interactive Euler Diagrams from Region Connection Calculus Specifications

François Schwarzentruber

Drawing Interactive Euler Diagrams from Region Connection Calculus Specifications

François Schwarzentruber

November 13, 2015

Abstract

This paper describes methods for generating interactive Euler diagrams. User interaction is needed to improve the aesthetic quality of the drawing without writing tedious formal specifications. More precisely, the user can modify the diagram's layout on the fly by mouse control. We prove that the satisfiability problem is in PSPACE and we provide two syntactic fragments such that the corresponding restricted satisfiability problem is already NP-hard. We describe (1) an improved local search based approach, (2) a method inspired from the gradient method and a hybrid method mixing both (1) and (2). A software tool was implemented and its implementation is described. We also experimentally compare the different methods. We first see that the improved local search and the hybrid method outperforms the local search from the literature and the gradient method for generating a diagram. Concerning interaction, the local search approach is not suitable but hybrid method and gradient method give both good results in terms of quality of drawings and stability. Specifications are written using region connection calculus (**RCC-8**), radius constraints and disjunctions. Euler diagrams are described as set of circles.

1 Introduction

Euler diagrams are pictures to understand relations between sets representing some concepts. In such pictures, sets (= concepts) are represented as regions in the plane and inclusions or intersections of those regions respectively depict inclusions or intersections of the corresponding sets. Euler diagrams are used in a wide range of application areas: medical data, biosciences, classification [21].

For instance, Figure 1 shows an Euler diagram representing the relations between the complexity classes P, NP, coNP, PSPACE, EXPTIME, NEXPTIME, coNEXPTIME and decidable [16] as many computer scientists may believe they are. For instance, $\mathsf{P} \subseteq \mathsf{NP}$ is represented by the fact that the disk corresponding to P is included in the disk corresponding to NP.

Euler diagrams can be drawn in a generic image editing software or a dedicated software as SketchSet [34]. Nevertheless, for some applications, it is

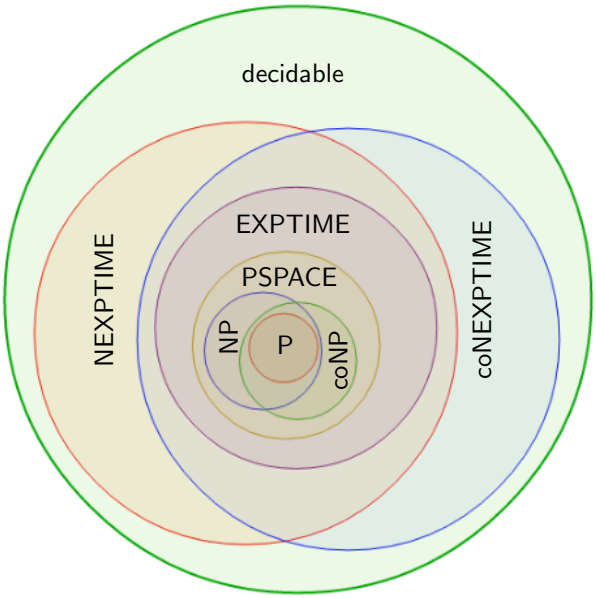
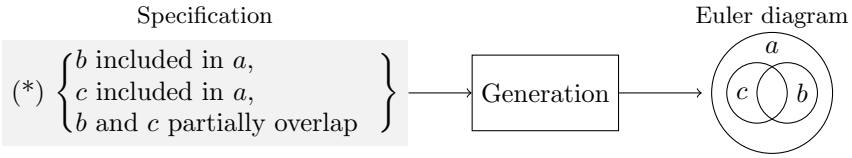


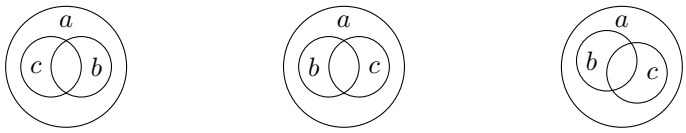
Figure 1: Euler diagram generated by our tool

convenient for the input to be an abstract specification of the Euler diagram. The diagram is then automatically generated from an abstract specification, for instance:



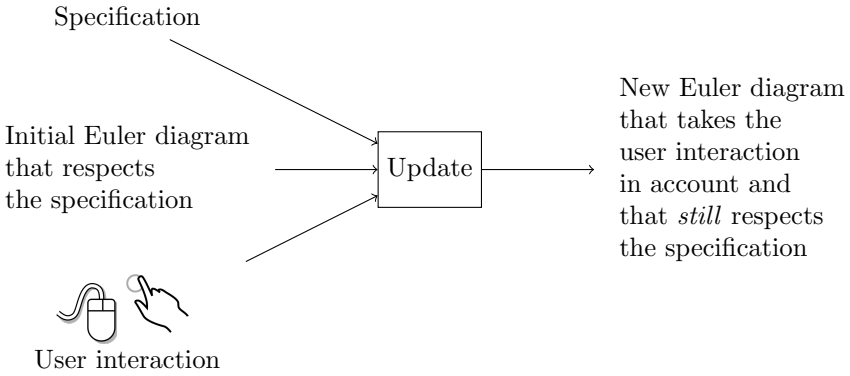
Automated generation is valuable. First, there is no effort spent to generate diagrams by hand and the generations can be applied to a huge quantity of diagrams. For instance, one may want to automatically generate Euler diagrams for several knowledge bases, e.g., knowledge bases are expressed in description logic ([3], [32]). Secondly the generation software itself certifies that generated Euler diagrams respect their specifications (or pinpoints which constraints are not satisfied in case the generated diagram does not fully satisfy to the specification).

Nevertheless, we claim that abstract specifications are not enough for specifying *nice* diagrams. For instance the following three Euler diagrams (i), (ii) and (iii) respect the same specification (*) given above.



(i) (ii) (iii)

Indeed, there are subjective requirements that are tedious to express as part of an abstract specification: for instance, one would prefer the Euler diagram (iii) to (i) and (ii) for aesthetic reasons. Therefore, we want generated Euler diagrams to be *interactive*, that is, we want to be able to move and resize circles of the diagram with a pointing device (mouse, etc.). More precisely, when interacting with a given Euler diagram, we want the initial Euler diagram to be updated so that it respects the user interaction while continuing to satisfy the abstract specification:



Furthermore, the drawing should be updated as smooth as possible with respect to the user interaction. We here refer to this feature as *stability* (we here borrow the vocabulary used for differential equations [12]). Technically, stability is measured by the distance between the initial drawing and the new drawing.

This paper presents a new version of the proof-of-concept software tool for drawing Euler diagrams by constraint solving with local search presented at JELIA 2014 [26].

Beyond the JELIA 2014 version, we give a new lower bound result for a fragment of the specification language (Proposition 3). We also significantly improved the algorithms for generation and user interaction.

- First, we improved the definition of the objective functions (see Subsection 4.1): now the value of an objective function represents the number of pixels in default for respecting a constraint.
- Secondly, we propose an improved version of local search (see Subsection 4.3): the more a circle is involved in unsatisfied constraints, the more it is inclined to be modified.

	Local search	Gradient method
Speed		✓
Few deadlocks	✓	
Stability		✓

Table 1: Advantages of the local search and gradient method

RCC-8-relations	Intuitive meanings
$DC(a, b)$	circles a and b are disconnected
$EC(a, b)$	circles a and b are externally connected
$PO(a, b)$	circles a and b partially overlap
$TPP(a, b)$	circle a is a tangential proper part of b
$TPP^{-1}(a, b)$	circle b is a tangential proper part of a
$NTPP(a, b)$	circle a is a non-tangential proper part of b
$NTPP^{-1}(a, b)$	circle b is a non-tangential proper part of a
$EQ(a, b)$	circles a and b are equal

Table 2: Intuitive meanings of **RCC-8**-relations

- We developed a new method based on the gradient method (see Section 5).
- We developed a hybrid method consisting in applying both gradient method and the improved version of local search alternatively (see Section 6).
- We also provide an experimental comparison of the methods (see Section 8). We claim here that the hybrid version combines the advantages of both the gradient method and the local search. The hybrid version is suitable for drawing Euler diagrams where interaction plays an important role.

On the one hand, gradient method is more efficient to find (local) minima than local search since the search is guided. On the other hand, if the new solution is not better, the drawing is no longer improved. We here refer to these bad behaviors as *deadlocks*. On the contrary, as local search neighborhoods are bigger, we have fewer deadlocks as with the gradient method. Table 1 sums the advantages of the two methods.

Region connection calculus [18] is a logic for reasoning about regions and topological constraints. It provides eight binary relations over regions (see Table 2 and Figure 2). Our proof-of-concept uses *region connection calculus* (**RCC-8**) as a specification language and sets of circles as models[1]. The reason is that semantics of **RCC-8**-predicates on circles are easily translated as objective functions used in the search algorithms (local search, etc.).

[1]The correct term should be 'disks' instead of 'circles' since a disk of radius r and center C also contain point whose distance from C are smaller than r. But we use here the term 'circles' as in our JELIA paper [26] and related work [30].

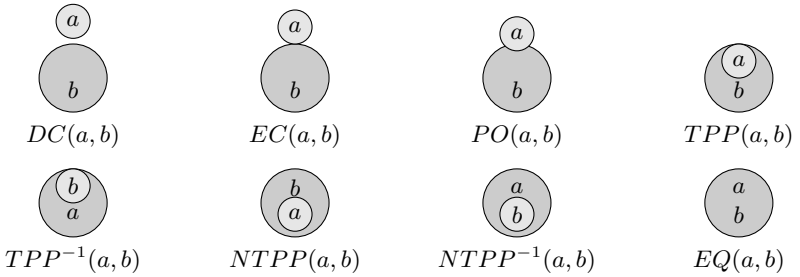


Figure 2: The eight **RCC-8**-relations in pictures

Interestingly, **RCC-8** provides topological features as 'sets a and b are externally connected'. It may be used to design Euler diagrams for math courses involving sets in a topological space[2] Note that **RCC-8** is less expressive than *abstract description* ([22], [30]) used by some other tools for generating Euler diagrams (see subsection 9.1).

The paper is organized as follows. In Section 2 we describe the specification language. Section 3 addresses the complexity of the satisfiability problem. Section 4 is dedicated to the improvement of the local search procedure. In Section 5, we present the variant of the gradient method. Section 7 presents the implementation of the hybrid method. Experimental results are presented in Section 8. Section 9 reviews related work. Perspectives are provided in the concluding Section 10.

2 Specification language

In this section, we present the language used to express constraints on circles in order to generate an Euler diagram and maintain an Euler diagram that satisfies the required abstract specification when the user interacts with it. It is the same language as in the first version of the tool [26]. The language proposes the eight predicates of *Region connection calculus* (**RCC-8**) [18].

2.1 Syntax

The syntax of the language $\mathcal{L}$ of constraints is defined by the following rule:

$$\varphi \quad ::= \quad R(a,b) \quad | \quad radius(a) = r \quad | \quad (\varphi \vee \varphi)$$

where a and b range over a set of constant symbols, r is a rational number and R ranges over the set of relation symbols of **RCC-8**

[2]For instance, let us consider the set of all bounded functions $f : \mathbb{R} \to \mathbb{R}^+$ and the topology defined by the uniform norm. Let b be the subset of functions bounded by 1 in the neighborhood of $\pm\infty$ and a be the subset of functions that converge to 0 in $\pm\infty$. As the boundary of b is the set of bounded functions that converge to 1 in $\pm\infty$, we should have the constraint $NTPP(a, b)$.

$\{DC, EC, PO, TPP, TPP^{-1}, NTPP, NTPP^{-1}, EQ\}$. Constant symbols denote circles. Intuitive meanings of **RCC-8**-relations are given in Table 2 and Figure 2. The construction $radius(a) = r$ is read as 'the radius of the circle a is r' and the construction $(\varphi_1 \vee \varphi_2)$ is read as 'φ_1 or φ_2'.

Example 1 *A specification corresponding to the drawing shown Figure 1 is the following set of formulas:*

$\{NTPP(\mathsf{P}, \mathsf{NP}),\ NTPP(\mathsf{P}, \mathsf{coNP}),\ NTPP(\mathsf{NP}, \mathsf{PSPACE}),$
$NTPP(\mathsf{PSPACE}, \mathsf{EXPTIME}),\ NTPP(\mathsf{EXPTIME}, \mathsf{NEXPTIME}),$
$NTPP(\mathsf{EXPTIME}, \mathsf{coNEXPTIME}),\ NTPP(\mathsf{NEXPTIME}, \mathsf{decidable}),$
$NTPP(\mathsf{coNEXPTIME}, \mathsf{decidable}),\ PO(\mathsf{NP}, \mathsf{coNP}),$
$PO(\mathsf{NEXPTIME}, \mathsf{coNEXPTIME}),\ radius(\mathsf{P}) = 30,\ radius(\mathsf{NP}) = 50,$
$radius(\mathsf{coNP}) = 50,\ radius(\mathsf{PSPACE}) = 80,\ radius(\mathsf{EXPTIME}) = 120,$
$radius(\mathsf{NEXPTIME}) = 180,\ radius(\mathsf{coNEXPTIME}) = 180,$
$radius(\mathsf{decidable}) = 250\}$.

Example 2 *Let us give two examples that illustrate the use of disjunctions:*

- *circle a is included in b:* $\{TPP(a, b) \vee NTPP(a, b)\}$*;*
- *borders of a and b intersect in a single point:*

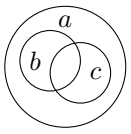
$$\{EC(a, b) \vee TPP(a, b) \vee TPP(b, a)\}.$$

2.2 Semantics

In this subsection, we explain how drawings are considered as first order models. Models are pairs $\mathcal{M} = \langle C, \sigma \rangle$ where:

- C is a non-empty set of circles of non-zero radius in the plane (for all $c \in C$, we respectively denote by $c.x$, $c.y$ and $c.r > 0$ the abscissa, the ordinate and the radius of the circle c ; we denote by $c.c$ the center of c);
- σ is an interpretation function that assigns to each constant symbol an element in C. $\sigma(a)$ is the circle named a in the model.

Example 3 *The drawing on the right is represented by the model $\mathcal{M} = \langle C, \sigma \rangle$ where*



- *C contains:*
  - *a circle c_a such that $c_a.x = 0$, $c_a.y = 0$, $c_a.r = 2$;*
  - *a circle c_b such that $c_b.x = -0.5$, $c_b.y = 0.2$, $c_b.r = 1$;*
  - *a circle c_c such that $c_c.x = 0.5$, $c_c.y = -0.2$, $c_c.r = 1$.*
- *and σ is such that $\sigma(a) = c_a$, $\sigma(b) = c_b$ and $\sigma(c) = c_c$.*

By abuse of notation, we write a instead of $\sigma(a)$ when no confusion can arise. Now let us define the truth conditions as follows. We define a relation $\mathcal{M} \models \varphi$ saying that the constraint φ is true in the model $\mathcal{M}$. In our context, $\mathcal{M} \models \varphi$ means that the Euler diagram $\mathcal{M}$ respects the constraint φ. In the following

definition, the number $d(a.c, b.c)$ is the Euclidean distance between the centers of a and b. Formally:

$$d(a.c, b.c) = \sqrt{(a.x - b.x)^2 + (a.y - b.y)^2}.$$

Definition 1 ()
Let $\mathcal{M} = \langle C, \sigma \rangle$ be a model. We define the relation $\mathcal{M} \models \varphi$ by induction on $\varphi \in \mathcal{L}$ as follows:

1.	$\mathcal{M} \models DC(a, b)$	iff $d(a.c, b.c) > a.r + b.r$;
2.	$\mathcal{M} \models EC(a, b)$	iff $d(a.c, b.c) = a.r + b.r$;
3.	$\mathcal{M} \models PO(a, b)$	iff $d(a.c, b.c) \in \left]\,\lvert a.r - b.r\rvert, a.r + b.r\,\right[$;
4.	$\mathcal{M} \models TPP(a, b)$	iff $d(a.c, b.c) = b.r - a.r$ (and $a.r \leq b.r$);
5.	$\mathcal{M} \models NTPP(a, b)$	iff $d(a.c, b.c) < b.r - a.r$ (and $a.r < b.r$);
6.	$\mathcal{M} \models TPP^{-1}(a, b)$	iff $d(a.c, b.c) = a.r - b.r$ (and $b.r \leq a.r$);
7.	$\mathcal{M} \models NTPP^{-1}(a, b)$	iff $d(a.c, b.c) < a.r - b.r$ (and $a.r > b.r$);
8.	$\mathcal{M} \models EQ(a, b)$	iff $a.c = b.c$ and $a.r = b.r$;
9.	$\mathcal{M} \models radius(a) = r$	iff $\sigma(a).r = r$;
10.	$\mathcal{M} \models (\varphi_1 \vee \varphi_2)$	iff $\mathcal{M} \models \varphi_1$ or $\mathcal{M} \models \varphi_2$.

Figure 3 shows the intervals where the quantity $d(a.c, b.c)$ belongs depending on the clause. For instance, the fact that a and b partially overlap corresponds to $d(a.c, b.c) \in \left]\,\lvert a.r - b.r\rvert, a.r + b.r\,\right[$ (clause 3). The equality $d(a.c, b.c) = a.r + b.r$ corresponds to the fact that a and b are externally connected (clause 2) and the equality $d(a.c, b.c) = |a.r - b.r|$ corresponds to the fact b is a tangential proper part of a (clause 4) (or a is a tangential proper part of b).

Example 4 *If $\mathcal{M}$ is the model defined in Example 3, then we have:*

$$\mathcal{M} \models PO(b, c), \qquad \mathcal{M} \models NTPP(b, a), \qquad \textit{and } \mathcal{M} \not\models DC(b, a).$$

3 Generation and satisfiability problem

In this section, we formally define the generation problem of an Euler diagram from a specification and the satisfiability problem which is the corresponding decision problem. We then recall some previous results [26] and new results concerning the lower bound complexity of the satisfiability problem.

3.1 Definitions

The problem we tackle here, called the *generation problem* is defined as follows:

- input: a finite set $I = \langle \varphi_1, \ldots, \varphi_n \rangle$ of constraints in $\mathcal{L}$;
- output: a model $\mathcal{M}$ such that for all $i \in \{1, \ldots, n\}$, $\mathcal{M} \models \varphi_i$.

The set I is also called a *specification*. The corresponding decision problem, called the *satisfiability problem* and noted $\mathcal{L}$-SAT takes the same input and outputs yes, if and only if there exists a model $\mathcal{M}$ such that for all $i \in \{1, \ldots, n\}$, $\mathcal{M} \models \varphi_i$:

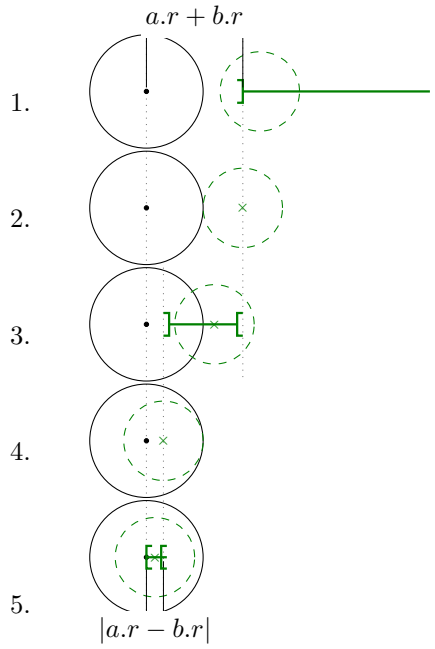
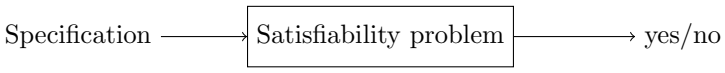


Figure 3: Intervals for $d(a.c, b.c)$ for clauses 1-5 of Definition 1



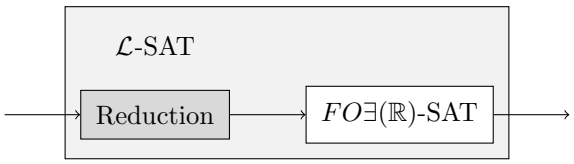
3.2 Upper bound

We recall the upper bound already given in [26] and we provide its proof.

Proposition 1 *$\mathcal{L}$-SAT is in PSPACE.*

Proof.

We polynomially reduce $\mathcal{L}$-SAT to the satisfiability problem of a formula in the existential fragment of the first order theory over real numbers $FO\exists(\mathbb{R})$ which is in PSPACE [4].



The reduction goes as follows. Let us consider an instance $I = \langle \varphi_1, \ldots, \varphi_n \rangle$. Let $c_1, \ldots, c_m$ be an enumeration of the constant symbols appearing in I. We

define χ_I to be the following $FO\exists(\mathbb{R})$-formula:

$$\exists c_1.x, \exists c_1.y, \exists c_1.r, \ldots, \exists c_m.x, \exists c_m.y, \exists c_m.r,\ tr(\varphi_1) \wedge \cdots \wedge tr(\varphi_n) \wedge \bigwedge_{i=1}^{m} (c_i.r > 0)$$

where tr is the translation of the constraints in the first order theory over real numbers given in Definition 1. More precisely:

$$\begin{array}{lcl}
tr\big(DC(a,b)\big) & = & d(a.c,b.c)^2 > (a.r+b.r)^2 \\
tr\big(EC(a,b)\big) & = & d(a.c,b.c)^2 = (a.r+b.r)^2 \\
tr\big(PO(a,b)\big) & = & (a.r-b.r)^2 < d(a.c,b.c)^2 \wedge \\
 & & \qquad d(a.c,b.c)^2 < (a.r+b.r)^2 \\
tr\big(TPP(a,b)\big) & = & d(a.c,b.c)^2 = (b.r-a.r)^2 \wedge a.r \leq b.r \\
tr\big(TPP^{-1}(a,b)\big) & = & d(a.c,b.c)^2 = (a.r-b.r)^2 \wedge b.r \leq a.r \\
tr\big(NTPP(a,b)\big) & = & d(a.c,b.c)^2 < (b.r-a.r)^2 \wedge a.r < b.r \\
tr\big(NTPP^{-1}(a,b)\big) & = & d(a.c,b.c)^2 < (a.r-b.r)^2 \wedge a.r > b.r \\
tr\big(EQ(a,b)\big) & = & a.x = a.x \wedge a.y = b.y \wedge a.r = b.r \\
tr\big(radius(a) = r\big) & = & a.r = r \\
tr\big((\varphi_1 \vee \varphi_2)\big) & = & tr(\varphi_1) \vee tr(\varphi_2)
\end{array}$$

where $d(a.c,b.c)^2$ stands for $(a.x - b.x)^2 + (a.y - b.y)^2$. For instance, $tr(DC(a,b) \vee EC(a,b))$ is the following first order formula

$$(d(a.c,b.c)^2 > (a.r+b.r)^2) \vee (d(a.c,b.c)^2 = (a.r+b.r)^2$$

and if $I = \langle EC(a,c), DC(a,b) \vee EC(a,b)\rangle$, then

$$\begin{array}{ll}
\chi_I = & \exists a.x, \exists a.y, \exists a.r \exists b.x, \exists b.y, \exists b.r \exists c.x, \exists c.y, \exists c.r \\
 & ((d(a.c,c.c)^2 = (a.r+c.r)^2) \wedge \\
 & \big((d(a.c,b.c)^2 > (a.r+b.r)^2) \vee (d(a.c,b.c)^2 = (a.r+b.r)^2)\big) \wedge \\
 & a.r > 0 \wedge b.r > 0 \wedge c.r > 0.
\end{array}$$

For any φ, $tr(\varphi)$ can be computed in polynomial time in the length of φ (see definition of tr above). Thus, computing χ_I from I can be done in polynomial time. Therefore, we have a procedure in polynomial space for solving $\mathcal{L}$-SAT. ■

3.3 Lower bound

In [26], it was already proven that $\mathcal{L}$-SAT is NP-hard. Proposition 2 restates more precisely that result by exhibiting explicitly the syntactic fragment of $\mathcal{L}$ used in the proof. Then Proposition 3 gives a new syntactic fragment of $\mathcal{L}$ for which the satisfiability problem is also NP-hard.

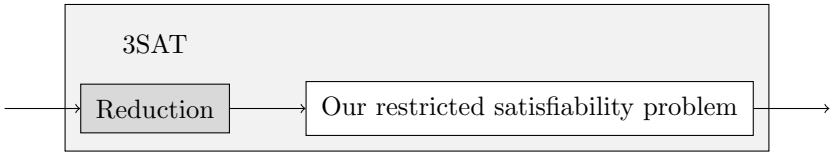
Proposition 2 *The satisfiability problem where instances does not contain any construction $radius(a) = r$ and such that we allow disjunctions only on clauses of the form:*

- $EC(\mathsf{ref}, a) \vee TPP(\mathsf{ref}, b)$*;*
- *clauses with three disjuncts of the form $EC(\mathsf{ref}, a)$ or $TPP(\mathsf{ref}, b)$*

where ref *is a given circle symbol and* a, b *are arbitrary circle symbols, is* NP*-hard.*

PROOF.

Let us polynomially reduce 3SAT (the satisfiability problem of a propositional formula in conjunction normal form where each clause has three literals) to our restricted satisfiability problem.



Let us consider an instance of 3SAT: $\chi = \bigwedge_{i=1}^{k} \ell_1^i \vee \ell_2^i \vee \ell_3^i$ where ℓ_j^i are literals, i.e., propositions or negations of a proposition. We construct an instance I_χ of our restricted satisfiability problem as follows. For all propositions p appearing in χ, we introduce a new constant symbol c_p. We also introduce an extra new constant symbol denoted by ref. Intuitively:

- p is true is encoded by $EC(\mathsf{ref}, c_p)$;
- p is false is encoded by $TPP(\mathsf{ref}, c_p)$.

The instance I_χ is the set that contains the following constraints:

- $EC(\mathsf{ref}, c_p) \vee TPP(ref, c_p)$ for all propositions p;
- $tr(\ell_1^i) \vee tr(\ell_2^i) \vee tr(\ell_3^i)$ for all $i \in \{1, \ldots, k\}$ where $tr(p) = EC(\mathsf{ref}, c_p)$ and $tr(\neg p) = TPP(\mathsf{ref}, c_p)$ for all propositions p.

We claim that the formula χ is satisfiable if and only if I_χ is a positive instance of our restricted satisfiability problem.

$\boxed{\Rightarrow}$ Let ν be a valuation such that $\nu \models \chi$. Let $\mathcal{M} = \langle C, \sigma \rangle$ where C contains:

- a circle c_{ref} such that $c_{\mathsf{ref}}.x = 0$, $c_{\mathsf{ref}}.y = 0$, $c_{\mathsf{ref}}.r = 100$;
- a circle $c_\top$ such that $c_\top.x = 150$, $c_\top.y = 0$, $c_\top.r = 50$;
- a circle $c_\bot$ such that $c_\bot.x = 50$, $c_\bot.y = 0$, $c_\bot.r = 50$.

and where σ is such that $\sigma(\mathsf{ref}) = c_{\mathsf{ref}}$ and for all propositions p,

$$\sigma(c_p) = \begin{cases} c_\top \text{ if } \nu \text{ makes } p \text{ true} \\ c_\bot \text{ if } \nu \text{ makes } p \text{ true.} \end{cases}$$

We have $\mathcal{M} \models I_\chi$.

$\boxed{\Leftarrow}$ Let $\mathcal{M}$ be a model such that $\mathcal{M} \models I_\chi$. We define a valuation ν as follows. For all propositions p,

- ν makes true p if $\mathcal{M} \models EC(\mathsf{ref}, c_p)$;
- ν makes false p if $\mathcal{M} \models TPP(\mathsf{ref}, c_p)$.

The valuation ν is well-defined by definition of I_χ and because a model can not satisfy both $EC(\mathsf{ref}, c_p)$ and $TPP(\mathsf{ref}, c_p)$ at the same time (recall that ref and c_p have strictly positive radius). We have $\nu \models \chi$.

Furthermore, I_χ can be computed in polynomial time in the length of χ. Hence, as 3SAT is NP-hard, our restricted satisfiability problem is NP-hard. ■

Proposition 3 *We also have NP-hardness by only using the EC predicate and the radius predicate and only one occurrence of the DC predicate.*

Proof.

Similarly to the proof of Proposition 2, let us still polynomially reduce 3SAT to $\mathcal{L}$-SAT. Let us consider an instance of 3SAT: $\chi = \bigwedge_{i=1}^{k} \ell_1^i \vee \ell_2^i \vee \ell_3^i$ where ℓ_j^i are literals, i.e., propositions or negations of a proposition. We construct an instance I_χ of our restricted satisfiability problem as follows. For all propositions p appearing in χ, we introduce a new constant symbol c_p. We also introduce the following new extra constant symbols: F, T, a, b and c. Intuitively:

- p is true is encoded by $EC(T, c_p)$;
- p is false is encoded by $EC(F, c_p)$.

The instance I_χ is the set that contains the following constraints:

- $radius(T) = 100$, $radius(F) = 100$, $radius(a) = 100$, $radius(b) = 100$;
- $EC(T, c)$, $EC(a, c)$, $EC(b, c)$, $EC(F, c)$;
- $EC(T, a)$, $EC(a, b)$, $EC(b, F)$;
- $DC(T, b)$;
- for all propositions p, $radius(c_p) = 50$;
- for all propositions p, $EC(T, c_p) \vee EC(F, c_p)$;
- $tr(\ell_1^i) \vee tr(\ell_2^i) \vee tr(\ell_3^i)$ for all $i \in \{1, \ldots, k\}$ where $tr(p) = EC(T, c_p)$ and $tr(\neg p) = EC(F, c_p)$ for all propositions p.

We claim that the formula χ is satisfiable if and only if I_χ is a positive instance of our restricted satisfiability problem.

$\boxed{\Rightarrow}$ Let ν be a valuation such that $\nu \models \chi$. Let $\mathcal{M} = \langle C, i \rangle$ where C contains:

- a circle c_c such that $c_c.x = 0$, $c_c.y = 0$, $c_c.r = 100$;
- a circle c_T such that $c_{true}.x = 200$, $c_{true}.y = 0$, $c_{true}.r = 100$;
- a circle c_F such that $c_F.x = -200$, $c_F.y = 0$, $c_F.r = 100$;
- a circle c_a such that $c_a.x = -100$, $c_a.y = 100\sqrt{3}$, $c_a.r = 100$;
- a circle c_b such that $c_b.x = 100$, $c_b.y = 100\sqrt{3}$, $c_b.r = 100$;
- a circle c_{true} such that $c_{true}.x = -350$, $c_{true}.y = 0$, $c_{true}.r = 50$;

- a circle c_{false} such that $c_{false}.x = 350$, $c_{false}.y = 0$, $c_{false}.r = 50$;

and where σ is such that $\sigma(a) = c_a$, $\sigma(b) = c_b$, $\sigma(c) = c_c$, $\sigma(T) = c_T$, $\sigma(F) = c_F$ and for all propositions p,

$$\sigma(c_p) = \begin{cases} c_{true} \text{ if } \nu \text{ makes } p \text{ true} \\ c_{false} \text{ if } \nu \text{ makes } p \text{ true.} \end{cases}$$

We have $\mathcal{M} \models I_\chi$.

$\boxed{\Leftarrow}$ Let $\mathcal{M}$ be a model such that $\mathcal{M} \models I_\chi$. We define a valuation ν as follows. For all propositions p,

- ν makes true p if $\mathcal{M} \models EC(T, c_p)$;
- ν makes false p if $\mathcal{M} \models EC(F, c_p)$.

The definition of ν is correct by definition of I_χ. The important argument is that $\mathcal{M}$ can not make both $EC(T, c_p)$ and $EC(F, c_p)$ true at the same time. Indeed, suppose by contradiction that $EC(T, c_p)$ and $EC(F, c_p)$ are true. Then $d(T.c, F.c) \leq 300$. But, as the model satisfy I_χ, the geometrical configuration of T, c, F, a, b makes that triangles $(T.c, a.c, c.c)$, $(a.c, c.c, b.c)$, $(c.c, b.c, F.c)$ are equilateral and distinct. It imposes points $T.c$, $c.c$ and $F.c$ to be aligned and $d(T.c, F.c) = d(T.c, c.c) + d(c.c, F.c) = 200 + 200 = 400$. This contradicts $d(T.c, F.c) \leq 300$. So, $EC(T, c_p)$ and $EC(F, c_p)$ can not be true at the same time. And we have, $\nu \models \chi$.

Furthermore, I_χ can be computed in polynomial time in the length of χ. Hence, as 3SAT is NP-hard, our restricted satisfiability problem is NP-hard.

For example, the instance 3SAT $p \vee q \vee q$ is translated by the following constraints (expressed here as an input of our tool):

```
circle("T");
circle("F");
circle("c");
circle("a");
circle("b");

addConstraint(EC("T", "c"));
addConstraint(EC("a", "c"));
addConstraint(EC("b", "c"));
addConstraint(EC("F", "c"));
addConstraint(EC("T", "a"));
addConstraint(EC("a", "b"));
addConstraint(EC("b", "F"));

addConstraint(DC("T", "b"));

addConstraint(radius("c", 100));
addConstraint(radius("T", 100));
addConstraint(radius("a", 100));
addConstraint(radius("b", 100));
addConstraint(radius("F", 100));

circle("cp");
circle("cq");

addConstraint(radius("cp", 50));
addConstraint(radius("cq", 50));
```

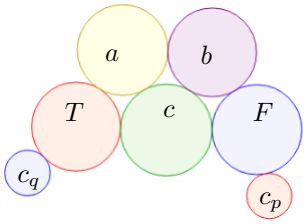


Figure 4: Model for the translation in the proof of Proposition 3 of the instance of 3SAT $p \vee q \vee q$

```
addConstraint(or(EC("T", "cp"), EC("F", "cp")));
addConstraint(or(EC("T", "cq"), EC("F", "cq")));

addConstraint(or(or(EC("T", "cp"), EC("T", "cq")), EC("T", "cq")));
```

Figure 4 shows a model for the translation in the proof of Proposition 3 of the instance of 3SAT $p \vee q \vee q$, that is, the specification above.

■

4 Local search

Given a problem instance $I = \langle \varphi_1, \ldots, \varphi_n \rangle$, we use a local search approach to determine an Euler diagram respecting the constraints of I. Generally speaking, local search constitutes a simple optimization approach which improves iteratively the current solution based on a neighborhood relation [16]. In our case, the local search algorithm explores the search space Ω of possible drawings $\mathcal{M}$ of a set of circles with the purpose of finding a feasible drawing satisfying the predicates (constraints) of the given formula.

In Subsection 4.1, we describe a new *soft semantics* used in the local search procedure. In Subsection 4.2, we describe the algorithm and in Subsection 4.3, we describe a new improvement not presented in [26].

4.1 Soft semantics

Usually in logic, semantics is given in terms of truth values as we saw in Subsection 2.2: a formula φ is either true or false in a given model. But, for the local search algorithm, we need the semantics to be *soft*: for each constraint φ, we design an *objective function* whose value is positive and the more its value is close to zero, the more the constraint is satisfied. In particular, if the value of the objective function is zero, then the constraint is fully satisfied.

A formula is evaluated according to an objective function $obj : \mathcal{L} \to \mathbb{R}$, defined by induction on φ as given in Table 3. Now, we interpret obj over a model $\mathcal{M}$ and we denote the evaluation value by $obj(\varphi)_{\mathcal{M}}$. Note that in the

Constraints φ	Objective functions $obj(\varphi)$
$DC(a,b)$	max(0, 1+(a.r + b.r) - d(a.c, b.c))
$EC(a,b)$	$\lvert d(a.c,b.c) - (a.r + b.r)\rvert$
$PO(a,b)$	$\lvert d(a.c,b.c) - max(a.r,b.r)\rvert$
$TPP(a,b)$	$\lvert d(a.c,b.c) - (b.r - a.r)\rvert$
$TPP^{-1}(a,b)$	$\lvert d(a.c,b.c) - (a.r - b.r)\rvert$
$NTPP(a,b)$	$max(0, d(a.c,b.c) + a.r - \alpha_{NTPP} \times b.r)$
$NTPP^{-1}(a,b)$	$max(0, d(a.c,b.c) + b.r - \alpha_{NTPP} \times a.r)$
$EQ(a,b)$	$d(a.c,b.c) + \lvert a.r - b.r\rvert$
$radius(a) = r$	$\lvert a.r - r\rvert$
$\varphi_1 \vee \varphi_2$	$min(obj(\varphi_1), obj(\varphi_2))$

where $\alpha_{NTPP} = 0.95$.

Table 3: Objective functions

constraint for $NTPP(a,b)$, the constant α_{NTPP} represents the ratio of radius of the outer circle b defining the region where the inner circle a should be.

The objective functions have been designed as follows. First, objective functions are designed such that its value represents the number of pixels in default. For instance, if $obj(EC(a,b)) = 2$, it means that circles a and b are distant of 2 or that circles a and b are too 2 pixel close. Secondly, when values of the objective functions are null then the corresponding constraint is satisfied. Formally:

Proposition 4 *If $obj(\varphi)_{\mathcal{M}} = 0$, then $\mathcal{M} \models \varphi$.*

Proof.

By induction on φ. Let us consider the basic cases $DC(a,b)$, $TPP(a,b)$ and $NTPP(a,b)$. Let $\mathcal{M}$ be a model.

Case $\varphi = DC(a,b)$. Suppose that $obj(DC(a,b))_{\mathcal{M}} = 0$. Thus,

$$max(0, 1 + (a.r + b.r) - d(a.c, b.c)) = 0.$$

It implies that $(a.r + b.r) < d(a.c, b.c)$. Hence, $\mathcal{M} \models DC(a,b)$.

Case $\varphi = TPP(a,b)$. If $obj(TPP(a,b))_{\mathcal{M}} = 0$, then $d(a.c,b.c) = (b.r - a.r)$, that is $\mathcal{M} \models TPP(a,b)$.

Case $\varphi = NTPP(a,b)$. Suppose that $obj(NTPP(a,b))_{\mathcal{M}} = 0$. Both $|d(a.c,b.c) - \frac{(b.r-a.r)}{2}|$ and $max\left(0, 0001 + \frac{a.r-b.r}{b.r}\right)$ are positive thus are zero. $0001 + \frac{a.r-b.r}{b.r} \leq 0$ implies $b.r > a.r$. On the other hand, $|d(a.c,b.c) - \frac{(b.r-a.r)}{2}| = 0$. It implies $d(a.c,b.c) = \frac{(b.r-a.r)}{2} < b.r - a.r$. Hence $\mathcal{M} \models NTPP(a,b)$.

Case $\varphi = \varphi_1 \vee \varphi_2$. The inductive case goes as follows. Suppose the proposition is true for φ_1 and φ_2. Let us prove that the proposition is true for $\varphi_1 \vee \varphi_2$. Suppose that $obj(\varphi_1 \vee \varphi_2)_{\mathcal{M}} = min(obj(\varphi_1)_{\mathcal{M}} = 0, obj(\varphi_2)_{\mathcal{M}}) = 0$. It implies that either $obj(\varphi_1)_{\mathcal{M}} = 0$ or $obj(\varphi_2)_{\mathcal{M}} = 0$. That is either $\mathcal{M} \models \varphi_1$ or $\mathcal{M} \models \varphi_2$. To conclude, either $\mathcal{M} \models \varphi_1 \vee \varphi_2$. ■ Note that those objective functions are established experimentally and prove to be appropriate to guide the local search algorithm described in the next subsection.

4.2 Algorithm

The pseudo-code is defined as follows:

```
M := generate randomly a drawing
while true do
    M_new := getSolutionInNeighborhood(M)
    if M_new is better than M then
        M := M_new
    endIf
endWhile
```

The algorithm never stops and keeps improving the current solution $\mathcal{M}$. To represent a model $\mathcal{M}$ (i.e., a drawing), $\mathcal{M}$ is considered as a vector, where indices are constant symbols c and each element $\mathcal{M}[c]$ is a circle represented by its center $(\mathcal{M}[c].x, \mathcal{M}[c].y)$ and its radius $\mathcal{M}[c].r$.

The function $getSolutionInNeighborhood(\mathcal{M})$ returns a new solution $\mathcal{M}_{new}$, where for all constant symbols c, $\mathcal{M}_{new}[c].x$, $\mathcal{M}_{new}[c].y$ and $\mathcal{M}_{new}[c].r$ are respectively obtained by adding randomly chosen numbers in an interval $[-\epsilon, \epsilon]$ to $\mathcal{M}[c].x$, $\mathcal{M}[c].y$ and $\mathcal{M}[c].r$. That is, a new drawing is obtained by moving every circle center from its current position to a new position and modifying slightly each radius (this move operator defines thus the neighborhood relation of our local search algorithm).

Solutions are compared with the following total order.

Definition 2 ()
Given two candidate solutions (drawings) $\mathcal{M}, \mathcal{M}_{new} \in \Omega$,

$$\mathcal{M}_{new} \text{ is better than } \mathcal{M}$$
$$\text{if}$$
$$\sum_{i=1}^{n} obj(\varphi_i)_{\mathcal{M}_{new}} \leq \sum_{i=1}^{n} obj(\varphi_i)_{\mathcal{M}},$$

where $obj(\varphi_i)_{\mathcal{M}_{new}}$ and $obj(\varphi_i)_{\mathcal{M}}$ are the values of the objective function $obj(\varphi_i)$ that corresponds to the i^{th} constraint φ_i for respectively $\mathcal{M}_{new}$ and $\mathcal{M}$.

Also here is the definition of a *good* drawing:

Definition 3 ()
The current drawing $\mathcal{M}$ is said to be *good* when

$$\sum_{i=1}^{n} obj(\varphi_i)_{\mathcal{M}} < 5$$

(the value 5 corresponds to 5 pixels of flaw in the satisfaction of constraints φ_i, see Section 8.4).

4.3 Improvement

We here present an improvement of the implementation presented in the JELIA 2014 paper [26] based on the following remark: intuitively, if all constraints involving circle a are satisfied, there are no reasons to modify circle a. Better, suppose that circle c is involved in constraint φ. The more φ is unsatisfied, the more we may modify circle c. Thus, now the choice of the ϵ is not uniform for all circles anymore. Now, we take ϵ to be proportional to $obj(\varphi)_{\mathcal{M}}$. More precisely, we take ϵ to be equal to $obj(\varphi)_{\mathcal{M}}$: we search for a drawing in the neighborhood of $\mathcal{M}$ by moving circles by number of pixels in the same order of magnitude than the number of pixels in default in the satisfaction of φ. Finally, we obtain the following implementation of $getSolutionInNeighborhood$:

```
function getSolutionInNeighborhood(M)
    M_new := M
    for all constraints φ ∈ I do
        for all circles c involved in φ do
            ε := obj(φ)_M
            M_new[c].x := M_new[c].x + rand([−ε, ε])
            M_new[c].y := M_new[c].y + rand([−ε, ε])
            M_new[c].r := M_new[c].r + rand([−ε, ε])
        endFor
    endFor
    return M_new
endFunction
```

where $rand([-\epsilon, \epsilon])$ chooses uniformly a number in $[-\epsilon, \epsilon]$, meaning that the probability of picking a number in $[\alpha, \beta]$ is $\frac{\beta-\alpha}{2\epsilon}$ for all α, β such that $-\epsilon \leq \alpha \leq \beta \leq \epsilon$.

5 Variant of the gradient descent

Contrary to local search, gradient descent [28] is an algorithm where the choice of the next solution around the current solution is guided directly by the global objective function. The next solution is computed from the current one by taking the opposite direction of the derivative of the objective function. For instance, in 1D, if the global objective function is $f(x) = x^3 - x^2$ and the current solution is 5 then, as $f'(x) = 3x^2 - 2x$ and as $f'(5) = 25$, the next candidate solution is $5 - 25\epsilon$ where ϵ is a positive real number. First, we describe the algorithm and then we describe how we implement our variant of our gradient descent.

5.1 Algorithm

The algorithm has essentially the same structure as for local search:

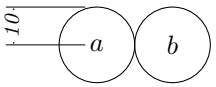
```
M := generate randomly a drawing
while true do
    |   M_new := getSolutionGradientVariant(M)
    |   if M_new is better than M then
    |    |   M := M_new
    |   endIf
endWhile
```

where $getSolutionGradientVariant$ is a function that computes a new solution (see the next subsection).

5.2 Deriving objective functions

Our global objective function to minimize is of the form $\sum_{i=1}^{n} obj(\varphi_i)$ where $\varphi_1, \ldots, \varphi_n$ are the constraints.

Example 5 *Suppose that the current drawing $\mathcal{M}$ is:*



Suppose that the specification is

$$\langle \varphi_1, \varphi_2, \varphi_3 \rangle = \langle EC(a,b), radius(a) = 50, radius(b) = 5 \rangle.$$

Let us use the notation:

$$\vec{\nabla} obj\big(\varphi\big)(\mathcal{M}) = \Big(\tfrac{\partial obj(\varphi)}{\partial a.x}(\mathcal{M}), \tfrac{\partial obj(\varphi)}{\partial a.y}(\mathcal{M}), \tfrac{\partial obj(\varphi)}{\partial a.r}(\mathcal{M}), \tfrac{\partial obj(\varphi)}{\partial b.x}(\mathcal{M}), \tfrac{\partial obj(\varphi)}{\partial b.y}(\mathcal{M}), \tfrac{\partial obj(\varphi)}{\partial b.r}(\mathcal{M})\Big)$$

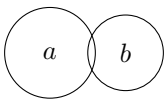
We have:

$$\begin{array}{lll}
\vec{\nabla} obj\big(EC(a,b)\big)(\mathcal{M}) & = (0, & 0, \quad 0, \quad 0, \quad 0, \quad 0) \\
\vec{\nabla} obj\big(radius(a) = 50\big)(\mathcal{M}) & = (0, & 0, \quad -1, \quad 0, \quad 0, \quad 0) \\
\vec{\nabla} obj\big(radius(b) = 5\big)(\mathcal{M}) & = (0, & 0, \quad 0, \quad 0, \quad 0, \quad 1)
\end{array}$$

If we write $\mathcal{M} = (\mathcal{M}[a].x, \mathcal{M}[a].y, \mathcal{M}[a].r, \mathcal{M}[b].x, \mathcal{M}[b].y, \mathcal{M}[b].r)$, then the new drawing $\mathcal{M}_{new}$ is:

$$\begin{array}{ll}
\mathcal{M}_{new} := \mathcal{M} & -\alpha \vec{\nabla} obj\big(EC(a,b)\big)(\mathcal{M}) \\
& -\beta \vec{\nabla} obj\big(radius(a) = 50\big)(\mathcal{M}) \\
& -\gamma \vec{\nabla} obj\big(radius(b) = 50\big)(\mathcal{M})
\end{array}$$

that is, we slightly augment the radius of a and b and $\mathcal{M}_{new}$ is:



The global objective function initial value was
$\sum_{i=1}^{3} obj(\varphi_i)_{\mathcal{M}} = 0 + 40 + 5 = 45$.
The global objective function current value is

$$\begin{aligned}\sum_{i=1}^{n} obj(\varphi_i)_{\mathcal{M}_{new}} &= |20 - ((10+\beta) + (10-\gamma))| + 40 - \beta + (5-\delta) \\ &= 45 - \beta - \gamma + |\beta - \gamma| < 45.\end{aligned}$$

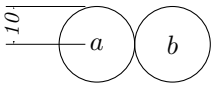
Formally, the computation of the new drawing is inspired from Subsection 4.3:

function *getSolutionGradientVariant*($\mathcal{M}$)
  $\mathcal{M}_{new} := \mathcal{M}$
  for all constraints $\varphi \in I$ **do**
    $\epsilon := obj(\varphi)_{\mathcal{M}}$
    $\mathcal{M}_{new} := \mathcal{M}_{new} - rand([-\epsilon, \epsilon]) \times \vec{\nabla} obj\big(\varphi\big)(\mathcal{M})$
  endFor
  return $\mathcal{M}_{new}$;
endFunction

where $rand([-\epsilon, \epsilon])$ chooses uniformly a number in $[-\epsilon, \epsilon]$.

Let us finish this subsection with an example where the gradient method does not work properly especially when the gradient directions are contradictory.

Example 6 *Suppose that the current drawing $\mathcal{M}$ is:*



Suppose that the specification is

$$\langle \varphi_1, \varphi_2, \varphi_3 \rangle = \langle EC(a,b), radius(a) = 50, radius(b) = 50 \rangle.$$

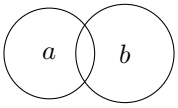
We have:

$$\begin{aligned}\vec{\nabla} obj\big(EC(a,b)\big)(\mathcal{M}) &= (0, \ 0, \ 0, \ \ 0, \ 0, \ 0) \\ \vec{\nabla} obj\big(radius(a) = 50\big)(\mathcal{M}) &= (0, \ 0, \ -1, \ 0, \ 0, \ 0) \\ \vec{\nabla} obj\big(radius(b) = 50\big)(\mathcal{M}) &= (0, \ 0, \ 0, \ \ 0, \ 0, \ -1)\end{aligned}$$

If we write $\mathcal{M} = (\mathcal{M}[a].x, \mathcal{M}[a].y, \mathcal{M}[a].r, \mathcal{M}[b].x, \mathcal{M}[b].y, \mathcal{M}[b].r)$, then the new drawing $\mathcal{M}_{new}$ is:

$$\begin{aligned}\mathcal{M}_{new} := \mathcal{M} \quad & -\alpha \vec{\nabla} obj\big(EC(a,b)\big)(\mathcal{M}) \\ & -\beta \vec{\nabla} obj\big(radius(a) = 50\big)(\mathcal{M}) \\ & -\gamma \vec{\nabla} obj\big(radius(b) = 50\big)(\mathcal{M})\end{aligned}$$

that is, we slightly augment the radius of a and b and $\mathcal{M}_{new}$ is:

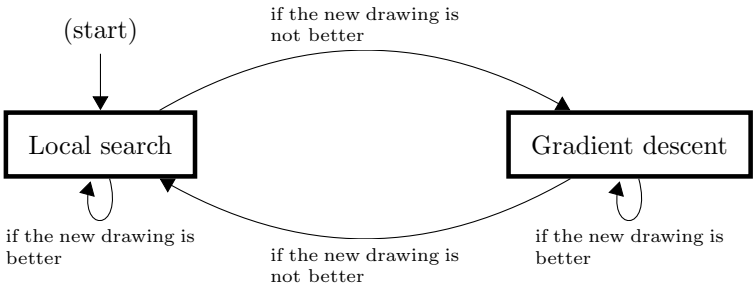


The global objective function initial value was
$\sum_{i=1}^{3} obj(\varphi_i)_{\mathcal{M}} = 0 + 40 + 40 = 80$.
The global objective function current value is
$\sum_{i=1}^{n} obj(\varphi_i)_{\mathcal{M}_{new}} = |20 - ((10 + \beta) + (10 + \gamma))| + 40 - \beta + 40 - \delta = 80$
and the quality of the drawing has not been improved by one step of our gradient method.

To avoid the drawback of our gradient method, we will design a method that combines local search and gradient method in the next section.

6 Hybrid method

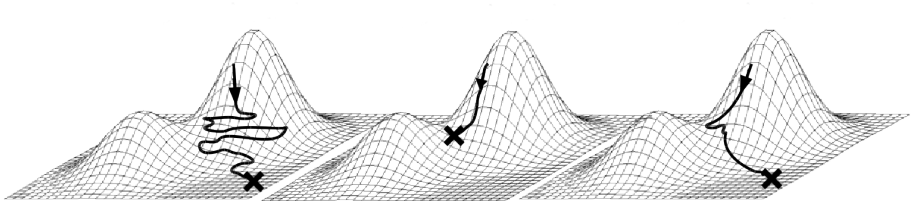
The hybrid method consists in alternating between applying local search (see Section 4 including the improvement described in Subsection 4.3) and applying our variant of gradient descent (see Section 5). As long as the current drawing is better than the previous one, we keep the current method and we switch when one method is not improving the current drawing, as explained in the following automaton:



It is designed to take advantage of both local search and gradient descent.

- In local search, the neighborhood is bigger than for gradient method (see Figure 5 (a)). When the neighborhood is big enough, we decrease the risk to reach a low-quality local minimum ([5], p. 298-299). Therefore, local search is by far much suitable for improving the quality of the solution than the gradient method.
- On the contrary, in the gradient method (see Figure 5 (b)), the neighborhood is more restricted therefore the method guides more the search process. Unfortunately we easily reach non global minimums.

As our hybrid method always switches between the two methods, we may expect that the method is faster than local search while reaching better solutions that gradient method. Figure 5 (c) tries to explain this fact.



(a) local search (b) gradient method (c) hybrid method

Figure 5: Search executions

7 Implementation

Our algorithm is implemented as a web application written in JavaScript. Figure 6 shows the graphical user interface of our tool.

7.1 Syntax used in the software

In the left part of the screen, the user adds a circle by writing `circle(name);` where `name` is a string for the name of the circle. Constraints are created with functions. For instance `TPP(name1, name2)` creates a TPP constraint between the circle named `name1` and the circle named `name2`. The construction `or(constraint1, constraint2)` returns a constraint that represents the disjunction of `constraint1` and `constraint2`. The construction `addConstraint(constraint)` adds `constraint` in the set of constraints. Figure 7 shows the specification of the Euler diagram shown in Figure 1. The user can add circles and constraints by clicking on the appropriate buttons in the palette.

7.2 Interaction

The user may *assist* the search algorithms (local search, gradient method or hybrid method). During the search, the user can move the circles by drag and drop and modify the radius of each circle. When the user makes a modification in the drawing, she directly modifies the current model $\mathcal{M}$. Those modifications are directly taken in account in real-time by the search algorithm.

Furthermore, the system guesses new potential constraints to add to the specification. For all pairs of circles (a, b), for all **RCC-8**-predicates R, it proposes the constraint $R(a, b)$ to be added to the guessed constraints when $obj(R(a, b))_{\mathcal{M}} < 5$ where 5 is a the threshold (see Definition 3).

7.3 Energy consumption

The implementation of algorithms given in Sections 4, 5 and 6 requires a high amount of computations and thus of energy. It just makes sure that your battery runs dry right before you can plug it in. To face this issue, the core of the **while** true loop is now temporized as follows:

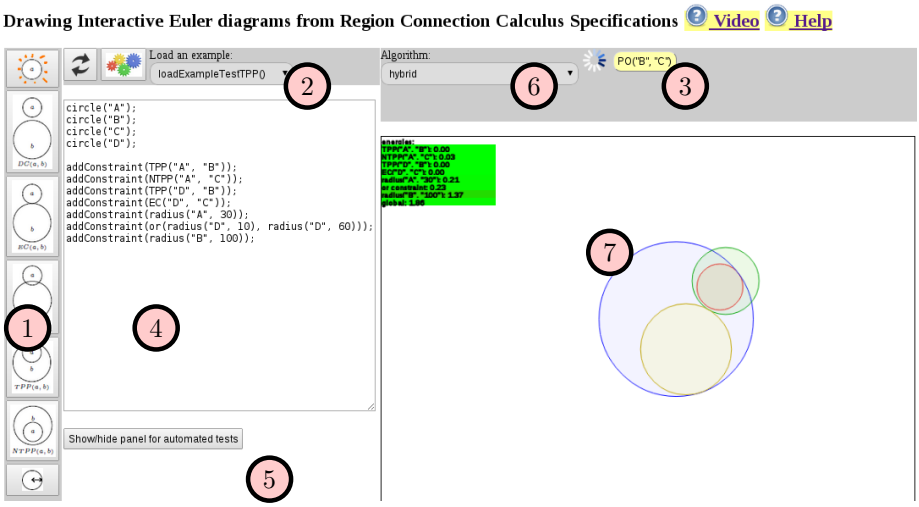


Figure 6: Graphical user interface. (1) the palette to add constraints. (2) controls to change the current example. (3) the guessed constraints from the interaction with the drawing. When the user moves/resizes a circle, the system guesses new constraints. For instance, if the user moves circle b so that circle b is outside circle a but touches circle a on the border, the system guesses the constraint $EC(a, b)$. (4) the current specification. (5) On the bottom left, the reader may reproduce experiments of Section 8. (6) controls to resume/pause the algorithm and to change the current strategy (gradient/local search/hybrid). (7) the drawing itself.

```
circle("P");              circle("NP");
circle("coNP");           circle("PSPACE");
circle("EXPTIME");        circle("NEXPTIME");
circle("coNEXPTIME");     circle("decidable");
addConstraint(NTPP("P", "NP"));
addConstraint(NTPP("P", "coNP"));
addConstraint(NTPP("NP", "PSPACE"));
addConstraint(NTPP("coNP", "PSPACE"));
addConstraint(NTPP("PSPACE", "EXPTIME"));
addConstraint(NTPP("EXPTIME", "NEXPTIME"));
addConstraint(NTPP("EXPTIME", "coNEXPTIME"));
addConstraint(NTPP("NEXPTIME", "decidable"));
addConstraint(NTPP("coNEXPTIME", "decidable"));
addConstraint(PO("NP", "coNP"));
addConstraint(PO("NEXPTIME", "coNEXPTIME"));
addConstraint(radius("P", 30));
addConstraint(radius("NP", 50));
addConstraint(radius("coNP", 50));
addConstraint(radius("PSPACE", 80));
addConstraint(radius("EXPTIME", 120));
addConstraint(radius("NEXPTIME", 180));
addConstraint(radius("coNEXPTIME", 180));
addConstraint(radius("decidable", 250));
```

Figure 7: Example of a specification

- if the drawing is not good (see Definition 3), the core of the **while** true loop is executed every milliseconds;
- if the drawing is good, the core of the **while** true loop is executed every seconds.

8 Experiments

In this section, we experiment both the generation of an Euler diagram from a specification and the interaction of the user. First, we describe how we generate benchmarks of Euler diagram specifications in Subsection 8.1. Then, we address the generation and the interaction in respectively Subsections 8.2 and 8.3. Note that experiments are fully reproducible in the software. We evaluate the hybrid method (Section 6), the original JELIA 2014 local search described in [26] (but with the new objective functions described in Table 3, see Section 4), the local search version including the improvement described in Subsection 4.3 (simply called local search) and the gradient method (Section 5).

8.1 Specification benchmark

Our generator function has two parameters: the number of circles taken as 6 and the number of constraints taken as 11.

- We generate 6 circles;

- We impose radius of circles to be between 20 and 150 (we do not impose exact constraints $radius(a) = r$ but conditions $\alpha \leq radius(a) \leq \beta$ where α, β are randomly uniformly chosen such that $20 \leq \alpha \leq \beta \leq 150$);
- We randomly select 11 tuples of circles (a, b) among the 6 circles and for each of these tuples (a, b), we add a constraint among $DC(a, b)$, $EC(a, b)$, $PO(a, b)$, $TPP(a, b)$, $NTPP(a, b)$ chosen uniformly;
- We then launch 3000 iterations of the hybrid method. If the final drawing is good (see Definition 3), we record the specification and the associated drawing in the benchmark.

The reader may think that 6 circles is not enough. Actually, we limit the number of circles and we specify a huge range for the radius so that many specifications are satisfiable. But, as we are interested in small specifications anyway, we believe that our benchmark is suitable for our purpose. We generated 1000 specifications (and drawings) as explained above and we stored them in benchmark.js available online.

8.2 Generation

For example for the specification of Figure 7, a good drawing (see Definition 3) is produced in approximately less than 0.5 seconds on a Pentium Dual-Core CPU 2.10 Ghz.

Algorithms for finding a drawing are not complete. We can find examples where the algorithm is stuck in a local minimum, for instance, example given in Figure 8. It shows a local minimum for the specification for the hybrid method. Nevertheless, when the user *assists* the algorithm, that is, when she moves some circles that obviously appear to be at wrong positions, the algorithm finds the global solution.

Now, let us evaluate the generation process, that is, we evaluate how many iterations we need to obtain a suitable drawing.

8.2.1 Experiments with our benchmark

We evaluate all the methods (hybrid, local search from JELIA 2014, local search and gradient) on the benchmark described in Subsection 8.1. The benchmark algorithm was just to produce specifications such that there exists a good drawing for it. Now for a fixed specification, we launch 3000 iterations of the method we want to test (hybrid, local search and gradient). Figure 9 shows an average (over 1000 specifications) of evolution of the energy, that is, the quantity $\sum_{i=1}^{n} obj(\varphi_i)_{\mathcal{M}_t}$ where t is the number of iterations and $\mathcal{M}_t$ is the drawing at step t.

Clearly, the gradient method is out of scope because it converges to non-global local minimums. On the contrary, local search, local search from JELIA

```
circle("A"); circle("B");  circle("C");
circle("D");  circle("E"); circle("F");
addConstraint(EC("A", "B"));  addConstraint(EC("B", "C"));
addConstraint(EC("C", "D"));  addConstraint(EC("D", "E"));
addConstraint(EC("E", "F"));   addConstraint(EC("F", "A"));
addConstraint(DC("A", "C"));  addConstraint(DC("B", "D"));
addConstraint(DC("B", "E"));  addConstraint(DC("A", "D"));
addConstraint(DC("C", "E"));  addConstraint(DC("A", "E"));
addConstraint(DC("B", "F"));  addConstraint(DC("C", "F"));
addConstraint(DC("D", "F"));
addConstraint(radius("A",30)); addConstraint(radius("B",30));
addConstraint(radius("C",30)); addConstraint(radius("D",30));
addConstraint(radius("E",30)); addConstraint(radius("F",30));
```

Figure 8: A specification, a non global local minimum (left) and global minimum (right)

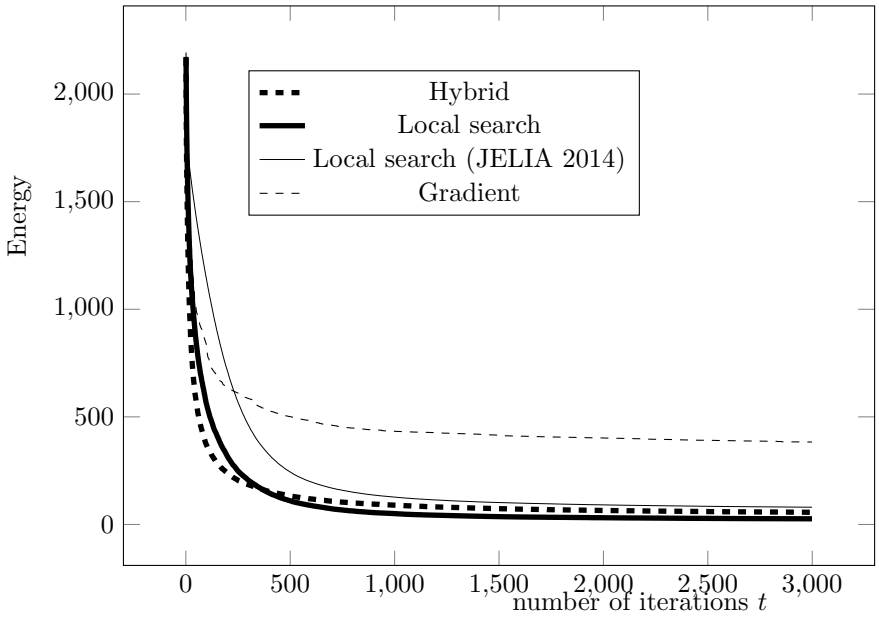


Figure 9: Average of evolution of the energy with the number of iterations for examples generated by the benchmark procedure

```
for t = 1 to 1000 do
    A.y := A.y + 1
    one iteration of of the evaluated method
endFor
```

Figure 10: Algorithm for evaluating interaction with a drawing

2014 and hybrid method are all suitable because they converge to better minimums in average. Figure 9 also shows that the improved local search searches gives a slightly better final drawing than the hybrid method and the JELIA 2014 local search in general. The following table reports the precise values of the energy at iteration $t = 3000$ in average for the four methods:

	Gradient	Local search (JELIA 2014)	**Local search**	Hybrid
Energy at iteration $t = 3000$ (in average)	383.02	80.13	**26.25**	56.55

Note that according to Figure 9 the hybrid method is faster at the beginning to give a suitable drawing. For instance, the following table gives how much iterations it requires for building a drawing whose energy is less that 200 in average:

	Gradient	Local search (JELIA 2014)	Local search	**Hybrid**
Number of iterations for generating a drawing of energy ≤ 200	(never)	594	308	**259**

8.3 Interaction

Concerning interaction, we evaluate the methods on the benchmark described in Subsection 8.1. Our experiment consists in simulating the use of the mouse on drawings. More precisely, depending on the method we test (local search, gradient method, hybrid method) we execute one of the following loops described in Figure 10.

In Figure 10, the instruction $A.y := A.y + 1$ simulates a move of circle A to the bottom. It simulates a drag and drop of circle A with a uniform speed to the bottom. After each move, we perform an iteration of local search (see Section 4) or the hybrid method (see Section 6). We note $\mathcal{M}_0$ the initial drawing and $\mathcal{M}_t$ is the drawing obtained after the execution of the t^{th} iteration of the for loop.

We are now interested in two aspects.

8.3.1 Respect of the constraints

The first aspect is the respect of the constraints φ_i. This is measured by the global measure $\sum_{i=1}^{n} obj(\varphi_i)$. More, precisely, for each iteration t, we are interested in the quantity $\sum_{i=1}^{n} obj(\varphi_i)_{\mathcal{M}_t}$.

8.3.2 Stability of the drawing

The second aspect is the stability of the drawing, that is the tendency of the drawing $\mathcal{M}_t$ to be close to the initial picture $\mathcal{M}_0$. To measure it, we propose to compute the square of the Euclidean norm between $\mathcal{M}_0$ and $\mathcal{M}_t$, noted $d^2(\mathcal{M}_0, \mathcal{M}_t)$.

Definition 4 ()
Given two drawings $\mathcal{M}_0$ and $\mathcal{M}_t$, we define $d^2(\mathcal{M}_0, \mathcal{M}_t)$ as the following quantity:

$$\sum_{\substack{\text{circles } c \text{ appearing} \\ \text{in a constraint } \varphi_i}} (\sigma_0(c).x - \sigma_t(c).x)^2 + (\sigma_0(c).y - \sigma_t(c).y)^2 + (\sigma_0(c).r - \sigma_t(c).r)^2$$

where σ_0 and σ_t are respectively the interpretation functions of $\mathcal{M}_0$ and $\mathcal{M}_t$.

8.3.3 Experimental results

We use the 1000 specifications generated by the procedure described in Subsection 8.1 as a benchmark. We also start with good drawings that are obtained from the same procedure described in Subsection 8.1. We then, for all methods (hybrid, local search and gradient) we run 1000 iterations where we moved the circle A as described in Figure 10 (1000 has the order of magnitude than the size of the drawing, so it corresponds to a typical interaction). Figure 11 reports an average over 1000 generated executions of the evolution $\sum_{i=1}^{n} obj(\varphi_i)_{\mathcal{M}_t}$ (y-axis) with respect to the number of the iteration t (x-axis). We experimentally observe that both gradient method and hybrid method correct the drawing more easily than local search. We see that the hybrid method is slightly better that the gradient method (see the graph on the bottom in Figure 11).

Figure 12 reports an average over 1000 executions of the evolution of $d^2(\mathcal{M}_0, \mathcal{M}_t)$ (y-axis) with respect to the number of the iteration t (x-axis). The measurements were made when we perform gradient method, local search iterations and hybrid method iterations. We experimentally observe that interaction gives drawing that are closer to the initial picture via gradient method/hybrid method than via local search. The gradient method is slightly better.

In conclusion, local search methods are clearly not suitable concerning interaction. But, the gradient method and the hybrid method are both suitable.

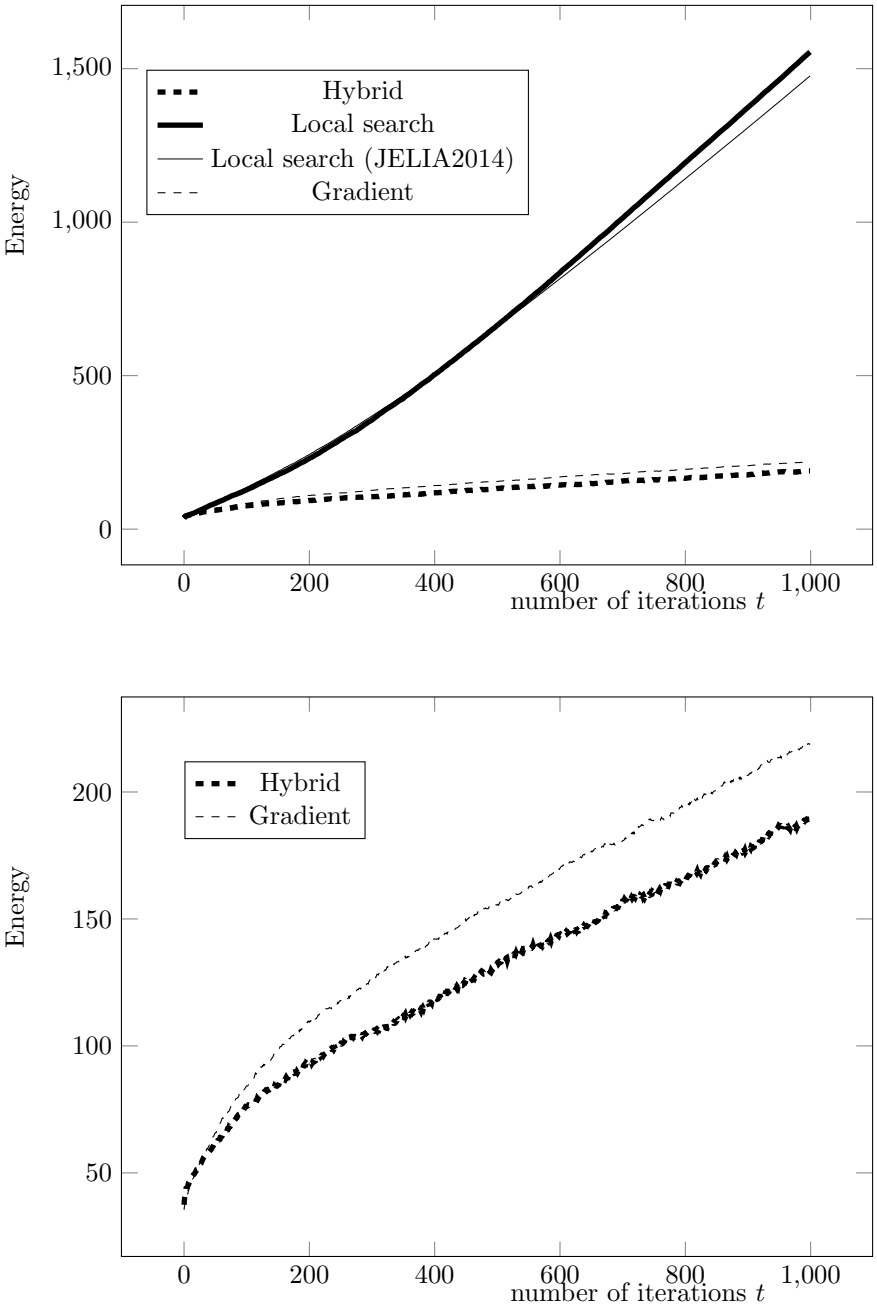


Figure 11: Average of the evolution of the global energy

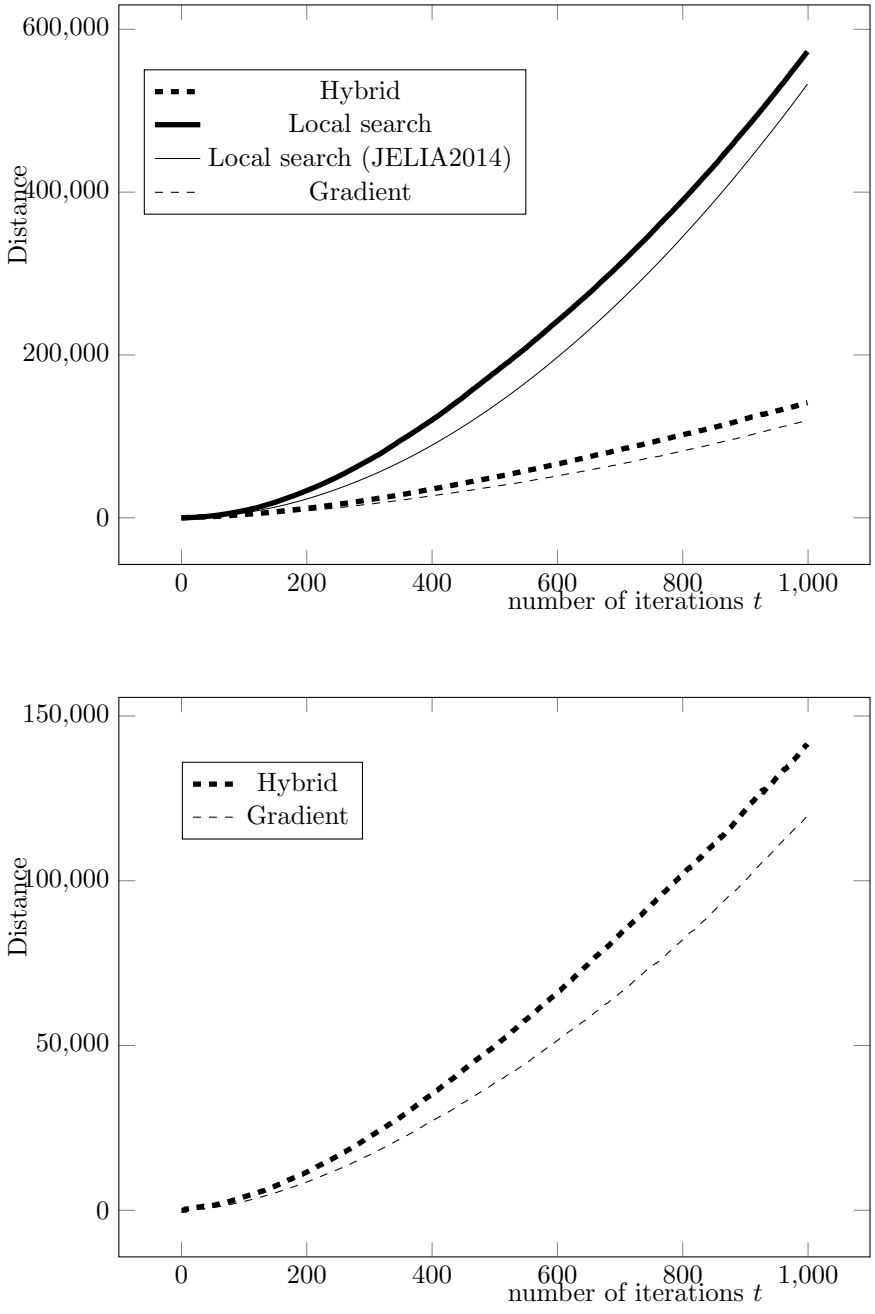


Figure 12: Average of the evolution of the distance to the initial drawing

Hybrid method is slightly more suitable concerning the global energy where the gradient method is more suitable when interested in stability.

8.4 User study

In order to evaluate the tool, we started a preliminary user study available online. In the graphical user interface, the respect of a constraint is represented with colors varying from red (the objective function value is greater than 5) to green (its value is 0). For most of the people, the feedback of the software is correct. It validates the definition of the objective functions (Table 3) and the definition of a *good drawing*.

In general, users prefer gradient method for generating diagrams. This contradicts Figure 9. It may be explained because the specifications users built by hand are easier to satisfy than specifications in the benchmark. The benchmark intentionally contains specifications difficult to satisfy.

Concerning interaction, the drawing correctly updates for most of the users. In general users prefer hybrid method. This confirms Figure 11.

9 Related work

9.1 Specification languages

9.1.1 Region connection calculus

Contrary to the concrete version of **RCC-8** we adopt in the present article (and in the JELIA 2014 paper [26]), variables are originally interpreted by non-empty regular closed regions of an abstract topological space [18] (a region X is said to be regular closed if X is equal to the closure of the interior of X). The satisfiability problem of a first order formula given in **RCC-8** is undecidable, more precisely not recursively enumerable [13].

An interesting fragment is the quantifier-free fragment of **RCC-8**. The corresponding satisfiability problem has been proven to be NP-complete [20]. Let $\mathcal{F}$ be a collection of non-empty regular closed regions of $\mathbb{R}^2$. We define the corresponding satisfiability problem $\mathcal{F}$-RSAT:

- input: a formula φ of the form $\exists x_1, \ldots \exists x_n, \bigwedge_{i,j \in \{1,\ldots n\}} \bigvee_{R \in C(i,j)} R(x_i, x_j)$ where n is a positive integer, $C(i,j)$ a subset of $\mathbb{REL}_{\mathbf{RCC\text{-}8}}$;
- output: yes iff there exists a model $\mathcal{M}$ where variables are interpreted as regions of $\mathcal{F}$ such that $\mathcal{M} \models \varphi$.

Interestingly, $\mathcal{F}$-RSAT is NP-complete when $\mathcal{F}$ is the set of disc-homeomorph regions of $\mathbb{R}^2$ ([23, 24]). In the current article, we proved that $\mathcal{F}$-RSAT is NP-hard and in PSPACE (see Propositions 1 and 2) when $\mathcal{F}$ is the set of closed disks of $\mathbb{R}^2$.

An extension of **RCC-8** with Boolean operations over sets has been studied in [11]. Soft semantics for **RCC-8** are also given in ([25, 29]).

9.1.2 Abstract description

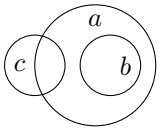
In some articles concerning the generation of Euler diagrams ([22], [30]), the language for the specification is called *abstract description* and is defined as follows.

Definition 5 ()
[30] An abstract description D is a pair $(L, \mathcal{Z})$ where

- L is a finite subset of constant symbols;
- $\mathcal{Z} \subseteq 2^L$ such that $\emptyset \in \mathcal{Z}$ and for all $c \in L$ there is $\Gamma \in \mathcal{Z}$ where $c \in \Gamma$.

The set L is a set of circle names and $\mathcal{Z}$ represents the zones in the diagram. Namely, if $\Gamma \in \mathcal{Z}$, Γ represents the set of points that are exactly in the disks of all elements in Γ. $\emptyset \in \mathcal{Z}$ represents the outside zone contained by no contours.

Example 7 $(\{a, b, c\}, \{\emptyset, \{c\}, \{c, a\}, \{a\}, \{a, b\}\})$ *is the abstract description corresponding to the drawing on the right.*



Note that not all abstract descriptions can be drawn with circles (without using repeated labels). For instance $(\{a, b, c, d\}, 2^{\{a,b,c,d\}})$ can not.

9.1.3 Comparing expressivity of RCC-8 and abstract descriptions

There are properties that can be expressed with an abstract description but not with **RCC-8**-constraints. Indeed, for instance, the abstract description $(L, \mathcal{Z})$ where

- $L = \{a, b, c\}$;
- $\mathcal{Z} = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}\}$.

has $\mathcal{M}_1$ as a model but not $\mathcal{M}'_1$ (see Figure 13). Note that $\mathcal{M}'_1$ is not a model for $(L, \mathcal{Z})$ because $\{a, b, c\} \notin \mathcal{Z}$. But there is no formula φ of the language $\mathcal{L}$ that is equivalent to the abstract description $(L, \mathcal{Z})$. Indeed, by contradiction, suppose there exists such a formula φ. Then, $\mathcal{M}_1$ is a model for φ. But, by the truth conditions (see Definition 1), $\mathcal{M}'_1$ is also a model for φ.

On the contrary, our language $\mathcal{L}$ is strictly more expressive since $\mathcal{M}_2$ and $\mathcal{M}'_2$ respect the same abstract description (that is, $D = (L, \mathcal{Z})$ where $L = \{a, b\}$ and $\mathcal{Z} = \{\emptyset, \{a\}, \{a, b\}\}$) whereas $NTPP(b, a)$ is true in $\mathcal{M}_2$ but not in $\mathcal{M}'_2$.

9.2 Interactive drawing

In this subsection, we sum up applications where users can interact with the drawing.

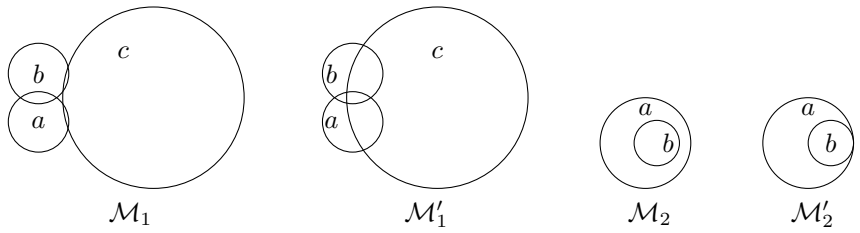


Figure 13: Models for comparing expressivity

9.2.1 Geometric software

In drawing software like Geogebra, one may state, for example, that Δ_1 contains point A and is orthogonal to line Δ_2 [10]. There are no interdependence between constraints and the positions of constrained objects to draw can be computed in polynomial time. The user can then move point A so that line Δ_1 still contains the point A.

9.2.2 Graphical interface

Similarly, in a graphical user interface library, the layout is computed from easily solvable constraints as 'the window is horizontally separated in two parts. The first part is a textbox. The second contains three buttons displayed vertically'. For these systems, various layout algorithms have been studied [2, 14]. In the same way, the user can resize windows etc.

9.2.3 Graph layout

Constraints have long been used for graph drawing. Finally, there exist tools to compute nice graphical representations of graphs [1, 6]. Displaying graphs consists in solving constraints such as two connected nodes are close and two different edges do not cross.

9.2.4 Interactive Visualization

The terminology 'interactive' is also used in interactive visualization. In that domain, it consists in displaying and querying huge database represented graphically. In particular, it consists in finding the best way to represent specific data ([7], [17]). As far as we know, the issue is not about displaying with respect to a set of constraints. On the contrary, in our work the adjective 'interactive' means that we can modify the drawing while respecting some constraints.

9.3 Bottom-up approach for drawing Euler diagrams

The visualization tool Tulip integrates a functionality for Euler diagrams [27]. The input of this system is given by an extensive description of the elements of

sets. For instance, the following can be a possible input:

$$\begin{array}{ll} \mathsf{P} & := \{path, linearprog\} \\ \mathsf{NP} & := \{path, linearprog, intlinearprog, sat\} \\ \mathsf{coNP} & := \{path, linearprog, intlinearprog, valid\} \end{array}$$

Tulip is a bottom-up approach. It considers the elements (in the example, elements are *path*, *linearprog*, etc.) as nodes in a graph constrained by the set-theoretical relations (in the example, $\mathsf{P} \subseteq \mathsf{NP}$, etc.). Tulip displays the graph and extracts an Euler diagram from it. The shape of a region corresponding to a set (for instance P) is delimited by the positions of the elements in that set (for instance, *path* and *linearprog*). Thus, the shape can be arbitrary and the diagram may be difficult to read. A similar approach can be found in [33].

On the contrary, our approach is top-down and deals with circles representing shapes of regions. We do not specify elements that are in sets. Furthermore, contrary to Tulip, our framework is suitable to capture constraints as 'the radius of the disc representing NP is $10cm$'.

9.4 Other top-down approaches for Euler diagrams

For generating Euler diagrams from abstract descriptions (see Subsection 9.1), one approach consists in generating a graph and find a planar layout of it ([8], [22]). Another approach for generating Euler diagrams with circles ([30], [31]) is based on the theory of piercings and enables to generate Euler diagrams in polynomial time. Very recently, another interesting tool is presented in [15] which is able to draw not only circles, but also ellipses. Yet, these approaches do not capture topological constraints as TPP (circle a is a tangential proper part of b) and the size of circles are not easily adjustable. Compared to these tools, our approach distinguishes itself by some interesting features. First, our tool is based on the **RCC-8** language. For instance, our tool allows the specification of topological constraints. Second, one can specify the radius of circles, and our system can then adjust dynamically these radii for a better visualization. Last but not least, in our approach, the user can always modify the drawing by moving and resizing circles and the system will adjust the drawing to respect the specification. The system will also guess and propose new constraints to add to the specification.

10 Conclusion

10.1 Summary

This study makes the bridge between logical framework **RCC-8**, generation of Euler diagrams (and more generally drawings under constraints), as well as heuristic search. We provide PSPACE membership of the satisfiability problem when regions are circles and some fragments that are NP-hard. We improved the local search algorithm presented in [26]. We also propose a stochastic version of

the gradient method. We created a hybrid method that combines the gradient method and the local search. Concerning the generation, gradient method is not suitable but both hybrid method and local search are suitable. Both the gradient method and the hybrid method offer good results for interaction with Euler diagrams but not local search. To sum up:

Tasks	Bad method(s)	Efficient method(s)
Generation	Gradient method	Local search $\sim$ Hybrid method
Interaction	Local search	Hybrid method $\sim$ Gradient method

More than Euler diagrams, this work could lead to the improvement of drawing tools (e.g. LibreOffice.draw) for architectures, scientists, artists.

10.2 Future work

A first extension is to add a large collection of elements in addition of circles (ellipses, rectangles, splines, etc.). It consists in extending the set of objective functions in Table 3. For instance, if a and b are rectangles (and rectangle a is described by $a.x1, a.y1, a.x2, a.x2$, etc.), then the semantics of $DC(a, b)$ is

$$a.x2 < b.x1 \text{ or } b.x2 < a.x1 \text{ or } a.y2 < b.y1 \text{ or } b.y2 < a.y1$$

and a possible objective function for $DC(a, b)$ is

$$min(a.x2 - b.x1, b.x2 - a.x1, a.y2 - b.y1, b.y2 - a.y1, 0).$$

If a is a circle and b is a rectangle, we can still define an objective function for constraints. Now, a challenge will be to handle elements where the number of parameters is unbounded, for instance splines where the number of key points is not known in advance. In particular, there may be a hidden constraint: minimize the number of key points in splines. It implicitely corresponds to the fact that the user may want to have simpliest geometrical shapes. The reader may imagine also constraints over shapes as 'a is circle or a is a rectangle'.

Then an interesting perspective is to combine constraints that do not require search (for instance constraints of Geogebra, or tractable fragments of **RCC-8** [20]) and constraints that require search. That is, the tool should be able to choose how to solve the constraints by detecting which method to apply and on which part of the drawing. This may impact both generation and interaction processes. In particular, it would be interesting to obtain a tight complexity result for the satisfiability problem of **RCC-8** over circles and to exhibit tractable fragments.

Another perspective is to improve the graphical interface. According to some few users (see Section 8.4), it is difficult to add constraints to a current drawing. We may imagine a graphical language that overlaps the drawing.

Finally, it would be interesting to add typical artificial intelligence features such as default reasoning [19]. For instance the sole constraint $TPP(\mathsf{P}, \mathsf{NP})$ (tangential proper part) should avoid the radius of P to be too small. This may be solved by using default reasoning: *by default*, $TPP(\mathsf{P}, \mathsf{NP})$ implies that the

radius of P is approximately the half of the radius of NP. Nevertheless, this extra constraint is relaxed as soon as it would provide inconsistency.

Concerning the generation, we may use complete algorithms, for instance we may use SMT (satisfiability modulo theories) solver [9] instead of heuristic search methods. We may also develop more efficient complete methods. For instance, complete algorithms such as those for generating Euler diagrams with circles ([30], [31]) based on the theory of piercings may be generalized to topological constraints or to other shapes.

But there are two main drawbacks of using complete algorithms. As far as we know, the generation will always fail if the specification is inconsistent whereas heuristic search may find a drawing that fits the constraint as much as possible. Secondly, interaction can not be implemented by exact methods since it is difficult to express that the new solution should both satisfy the specification and the constraint due to the interaction while be a close solution to the current previous one.

Another interesting research problem concerns the axiomatization. Is there an axiomatization of **RCC-8** where objects are circles? Having an axiomatization may help us to improve the software so that it could give explanations for the generated drawings. We may then imagine that the user can discuss with the system for designing a drawing.

Great, where can I find the tool? The tool, a video, a preliminary user study, experimental data and source files are available at

`http://people.irisa.fr/Francois.Schwarzentruber/constrainteddrawing/`

Acknowledgments. We still wish to thank the three JELIA reviewers for their critical comments and pointers to relevant studies. We also thank the reviewers of this journal version. We thank users who accepted to perform the user study. We thank also Benjamin Boutin for the example in footnote 2.

References

[1] David Auber. Tulip—a huge graph visualization framework. In *Graph Drawing Software*, pages 105–126. Springer, 2004.

[2] Alan Borning, Kim Marriott, Peter J. Stuckey, and Yi Xiao. Solving linear arithmetic constraints for user interface applications. In *ACM Symposium on User Interface Software and Technology*, pages 87–96, 1997.

[3] Jim Burton, Gem Stapleton, John Howse, and Peter Chapman. Visualizing concepts with Euler diagrams. In Tim Dwyer, Helen C. Purchase, and Aidan Delaney, editors, *Diagrammatic Representation and Inference - 8th International Conference, Diagrams 2014, Melbourne, VIC, Australia, July 28 - August 1, 2014. Proceedings*, volume 8578 of *Lecture Notes in Computer Science*, pages 54–56. Springer, 2014.

[4] John Canny. Some algebraic and geometric computations in pspace. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 460–467. ACM, 1988.

[5] Sanjoy Dasgupta, Christos H Papadimitriou, and Umesh Vazirani. *Algorithms*. McGraw-Hill, Inc., 2006.

[6] John Ellson, Emden R Gansner, Eleftherios Koutsofios, Stephen C North, and Gordon Woodhull. Graphviz and dynagraph—static and dynamic graph drawing tools. In *Graph Drawing Software*, pages 127–148. Springer, 2004.

[7] Jean-Daniel Fekete and Catherine Plaisant. Interactive information visualization of a million items. In *Information Visualization, 2002. INFOVIS 2002. IEEE Symposium on*, pages 117–124. IEEE, 2002.

[8] Jean Flower and John Howse. Generating Euler diagrams. In Mary Hegarty, Bernd Meyer, and N. Hari Narayanan, editors, *Diagrams*, volume 2317 of *Lecture Notes in Computer Science*, pages 61–75. Springer, 2002.

[9] John Harrison. *Handbook of practical logic and automated reasoning*. Cambridge University Press, 2009.

[10] Markus Hohenwarter and Judith Preiner. Dynamic mathematics with geogebra. *Journal of Online Mathematics and its Applications*, 7, 2007.

[11] Roman Kontchakov, Yavor Nenov, Ian Pratt-Hartmann, and Michael Zakharyaschev. On the decidability of connectedness constraints in 2d and 3d euclidean spaces. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 957, 2011.

[12] JP LaSalle. Stability theory for ordinary differential equations. *Journal of Differential Equations*, 4(1):57–65, 1968.

[13] Carsten Lutz and Frank Wolter. Modal logics of topological relations. *Logical Methods in Computer Science*, 2:1–14, 2006.

[14] Kim Marriott, Peter Moulder, Peter J. Stuckey, and Alan Borning. Solving disjunctive constraints for interactive graphical applications. In Toby Walsh, editor, *CP*, volume 2239 of *Lecture Notes in Computer Science*, pages 361–376. Springer, 2001.

[15] Luana Micallef and Peter Rodgers. eulerape: Drawing area-proportional 3-Venn diagrams using ellipses. *PLoS ONE*, 9(7):e101717, 07 2014.

[16] Christos H Papadimitriou. *Computational complexity*. John Wiley and Sons Ltd., 2003.

[17] Catherine Plaisant, Megan Monroe, Tamra Meyer, and Ben Shneiderman. Interactive visualization. *Big Data and Health Analytics*, page 243, 2014.

[18] David A. Randell, Zhan Cui, and Anthony G. Cohn. A spatial logic based on regions and connection. *KR*, 92:165–176, 1992.

[19] Raymond Reiter. A logic for default reasoning. *Artificial intelligence*, 13(1):81–132, 1980.

[20] Jochen Renz and Bernhard Nebel. On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the region connection calculus. *Artificial Intelligence*, 108(1):69–123, 1999.

[21] Peter Rodgers. A survey of Euler diagrams. *J. Vis. Lang. Comput.*, 25(3):134–155, 2014.

[22] Peter Rodgers, Leishi Zhang, and Andrew Fish. General Euler diagram generation. In *Diagrammatic Representation and Inference, 5th International Conference, Diagrams 2008, Herrsching, Germany, September 19-21, 2008. Proceedings*, pages 13–27.

[23] Marcus Schaefer, Eric Sedgwick, and Daniel Štefankovič. Recognizing string graphs in np. *Journal of Computer and System Sciences*, 67(2):365–380, 2003.

[24] Marcus Schaefer and Daniel Stefankovic. Decidability of string graphs. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 241–246. ACM, 2001.

[25] Steven Schockaert, Martine De Cock, and Etienne E Kerre. Spatial reasoning in a fuzzy region connection calculus. *Artificial Intelligence*, 173(2):258–298, 2009.

[26] François Schwarzentruber and Jin-Kao Hao. Drawing Euler diagrams from region connection calculus specifications with local search. In Eduardo Fermé and João Leite, editors, *Logics in Artificial Intelligence - 14th European Conference, JELIA 2014, Funchal, Madeira, Portugal, September*

24-26, 2014. Proceedings, volume 8761 of *Lecture Notes in Computer Science*, pages 582–590. Springer, 2014.

[27] Paolo Simonetto, David Auber, and Daniel Archambault. Fully automatic visualisation of overlapping sets. In *Computer Graphics Forum*, volume 28, pages 967–974. Wiley Online Library, 2009.

[28] Jan Snyman. *Practical mathematical optimization: an introduction to basic optimization theory and classical and new gradient-based algorithms*, volume 97. Springer, 2005.

[29] Muralikrishna Sridhar, Anthony G Cohn, and David C Hogg. From video to rcc8: exploiting a distance based semantics to stabilise the interpretation of mereotopological relations. In *Spatial Information Theory*, pages 110–125. Springer, 2011.

[30] Gem Stapleton, Leishi Zhang, John Howse, and Peter Rodgers. Drawing Euler diagrams with circles. In *Diagrammatic Representation and Inference, 6th International Conference, Diagrams 2010, Portland, OR, USA, August 9-11, 2010. Proceedings*, pages 23–38.

[31] Gem Stapleton, Leishi Zhang, John Howse, and Peter Rodgers. Drawing Euler diagrams with circles: The theory of piercings. *IEEE Trans. Vis. Comput. Graph.*, 17(7):1020–1032, 2011.

[32] Frank Van Harmelen, Vladimir Lifschitz, and Bruce Porter. *Handbook of knowledge representation*, volume 1. Elsevier, 2008.

[33] Anne Verroust and Marie-Luce Viaud. Ensuring the drawability of extended Euler diagrams for up to 8 sets. In Alan F. Blackwell, Kim Marriott, and Atsushi Shimojima, editors, *Diagrams*, volume 2980 of *Lecture Notes in Computer Science*, pages 128–141. Springer, 2004.

[34] Mengdi Wang, Beryl Plimmer, Paul Schmieder, Gem Stapleton, Peter Rodgers, and Aidan Delaney. Sketchset: Creating Euler diagrams using pen or mouse. In *2011 IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2011, Pittsburgh, PA, USA, September 18-22, 2011*, pages 75–82.